ISSN 0956-9979 AUGUST 1995

VIRUS BULLETIN

THE INTERNATIONAL PUBLICATION ON COMPUTER VIRUS PREVENTION, RECOGNITION AND REMOVAL

Editor: **Ian Whalley**

Assistant Editor: **Megan Palfrey**

Technical Editor: **Jakub Kaminski**

Consulting Editors:

Richard Ford, NCSA, USA
Edward Wilding, Network Security, UK

IN THIS ISSUE:

• **Israel today.** In Israel, use of 5.25-inch floppy disks is more widespread than that of 3.5-inch diskettes. This, among other considerations peculiar to the country, brings its own problems, as the article on p.16 discusses.

• **Speedy access.** 32-bit file access is designed to speed up *Windows for Workgroups* - but will it be a problem for anti-virus software developers? See our article on p.13.

• **Detecting a new way.** Pavel Baudiš, of *Alwil Software*, is a little-known figure in the anti-virus industry - the product he has helped to develop, however, scores extremely well in detecting polymorphic and other viruses. How did he get there? See p.6.

CONTENTS

EDITORIAL

Here Come the Little People

Browsing through the results of last month's DOS Scanner comparative reveals few surprises. The ranking of the major contenders in the anti-virus field are, by and large, as expected. However, one cliché springs immediately to mind - size is not everything.

Look at the polymorphic tests: the results from this section offer a reasonably accurate measure of the level of technological development in a product. Three of the top four products in this area are produced by small, relatively unknown, companies. *AVP* comes from a group in Moscow; *AVAST!*, from *Alwil* in the Czech Republic; and *AVScan*, from *H+BEDV* in Germany.

‘‘ you can bet your bottom dollar the (people who design the boxes) are paid more ’’

Curious, isn't it? They are all small companies, all based (both geographically and in terms of the anti-virus product market) out of the mainstream, and they produce products with extremely high detection rates. What could account for this strange inverse relationship between marketing budget and detection rate?

There does appear to be a connection here. Writing this editorial, I find myself making notes with a pen marked with the logo of an anti-virus company on a pad of paper emblazoned with the name of another. A third company is proud to produce the mat upon which my mouse sits, unused; a fourth, a pair of socks (which I've temporarily mislaid) and a set of dictionaries which has a prominent place on my desk; a fifth, a screwdriver with which I took my computer to bits yesterday. I have heard of, but do not have, other marketing freebies, such as sweat bands, little yellow footballs, and even rugby balls. I am sure there are many more.

None of this promotional material comes from the little people with the large detection rates. Some of it comes from the fairly big people with the almost-as-good detection rates, and quite a lot comes from the big people with the not-very-high detection rates.

Look also at the boxes containing the products from the better-known companies. Stylized cyborgs brandish shields, spanners of various colours levitate, a surgical syringe motif abounds - the gloss is almost dazzling. Who is valued more in such companies, the programmers and virus analysts, or the people who design the boxes? You can bet your bottom dollar the latter are paid more.

Now look at the boxes of the three products mentioned above. Two of these arrived with no more packaging than a buff envelope and some release notes. Herein lies the problem. Major corporates are unlikely to go for such products as their mainstream defence. Why?

These corporates feel (not unreasonably, and in many senses rightly) that they need the security of a larger, better-known company; or at least one which has a higher profile. Quite apart from the fact that a larger company will probably be able to provide wider coverage of the different operating systems in use in many modern computing environments, there is a feel-good factor at work; you can be reasonably sure that the company will not go under, will speak your language and be easily contactable - with any luck, in your own country.

The language issue is not insignificant. The review copies of all of the three products under discussion here were in English, but in all three cases, it is English obviously written by a non-native speaker. This too will matter to the corporate market; the first impression they will get is that the help is ‘peculiar’ in some way, because it does not read as easily as documentation produced by a native speaker. Whether consciously or unconsciously, this will reflect in the mind on the overall quality of the product.

To conclude; these days, when larger anti-virus companies use elaborate themed stands at computer shows, when (in one recent notable case), the chairman of one such company is interviewed in a high-profile glossy gentleman's magazine, it is gratifying to be reassured that the little people are still with us, following their own yellow brick road, trying to gain recognition and respect. If they continue in their current vein, they too will surely reach the Emerald City.

NEWS

The S&S of Time

Shortly after the last edition of *VB* went to press, *S&S* launched version 7.5 of their *Anti-Virus Toolkit* at the *Networks '95* show at the *NEC* in Birmingham. Amongst the new features incorporated are WinGuard and AHA.

WinGuard is a VxD protection system for *Windows* [*see More Bits for Your Buck, p.13*] which *S&S* claims will detect even highly-polymorphic viruses, as it uses the same scanning engine as FindVirus, the command-line scanner. Whilst the product does not use base memory in DOS boxes, the overall amount of memory available to *Windows* applications is decreased.

AHA stands for Advanced Heuristic Analysis. *S&S* says that this has an extremely low false positive rate (false positives are traditionally one of the major problems with heuristic scanners), and that it is able to detect 80% of viruses about which the scanner has no prior knowledge.

On the *NetWare* side, version 7.5 of the *Anti-Virus Toolkit NLM* features on-access scanning (otherwise known as real time scanning). This feature, which is common in NLMs, allows files to be scanned as they are placed upon, and retrieved from, the server.

S&S expects to ship a *Macintosh* product shortly; initially as part of corporate licences, but later as a stand-alone product. Scanners for *SCO UNIX*, *Windows NT* and *Windows 95* are planned for the near future - the first two should be released in August, the last in October.

In addition to these new anti-virus products, *Networks '95* also saw the launch of version 2.0 of *Dr Solomon's Audit*. The new version is said to allow comprehensive auditing of both software and hardware on a network, in addition to the configuration files. As with the *Windows NT* and *SCO UNIX* products, this will also ship in August ▮

Virus Prevalence Table - June 1995

Virus	Incidents	(%) Reports
Form	30	17.3%
AntiEXE	26	15.0%
Parity Boot	12	6.9%
AntiCMOS	11	6.4%
NYB	10	5.8%
Jumper	8	4.6%
Bupt	6	3.5%
Junkie	6	3.5%
Monkey.B	6	3.5%
Sampo	5	2.9%
Stoned.NoInt	5	2.9%
Amse	4	2.3%
Quicky	4	2.3%
Cascade-1701	3	1.7%
Telefonica	3	1.7%
V-Sign	3	1.7%
DA_Boys	2	1.2%
JackRipper	2	1.2%
Keypress-1216	2	1.2%
Natas	2	1.2%
One_Half	2	1.2%
Tequila	2	1.2%
*Other	19	11.0%
Total	173	100%

* The Prevalence Table includes one report of each of the following viruses: Stoned.Angelina, Barrotes, Cinderella-390, CMOS1, EXE_Bug.A, Frodo.Frodo.A, Mange-Tout, Mongolian, Monkey.A, Nomenklatura, PrintScreen.a, She_Has, Sibylle, Stoned.g, Stoned.p, Stoned.p2, Stoned.Stonehenge, Tai-Pan, Trackswap.

Windows 95: More of the Same?

The June 1995 issue of the *American Eagle Publications* newsletter '*Underground Technology Review*' carries source code for a virus designed for *Windows 95*.

This virus is part of a system presented by *American Eagle* to exploit the insecurities inherent in *Windows 95*. Shared data areas exist in that operating system, which allow data to be exchanged between tasks.

It uses viral code modelled on the Jerusalem virus; this code is used to infect EXE files with a keyboard monitor which passes data into a data area which is available to other tasks. This would allow a background task to capture keypresses from other applications, presumably whilst another user is logged in.

Other than exploiting the shared data areas (which is not necessary under DOS and *Windows*, as the whole machine is a shared data area), the source code does not appear to use any features which are specific to *Windows 95*. The fact that the viral part of the code is based on the Jerusalem virus supports this.

The ease with which such code can be modified to work on, if not to be completely integrated with, the new operating system is cause for concern. However, it appears that the virus under discussion is not without its limitations.

It can only monitor keystrokes in a session in which it has been able to install its Int 09h handler, which clearly implies DOS sessions. What would happen if the virus infects a full *Windows 95* executable is not clear ▮

IBM PC VIRUSES (UPDATE)

The following is a list of updates and amendments to the *Virus Bulletin Table of Known IBM PC Viruses* as of 21 July 1995. Each entry consists of the virus name, its aliases (if any) and the virus type. This is followed by a short description (if available) and a 24-byte hexadecimal search pattern to detect the presence of the virus with a disk utility or a dedicated scanner which contains a user-updatable pattern library.

Type Codes

C	Infects COM files	**M**	Infects Master Boot Sector (Track 0, Head 0, Sector 1)
D	Infects DOS Boot Sector (logical sector 0 on disk)	**N**	Not memory-resident
E	Infects EXE files	**P**	Companion virus
L	Link virus	**R**	Memory-resident after infection

_591 — **CN:** An appending, 591-byte direct infector containing a destructive payload. After April 1991, on the 26th day of every month it overwrites the DOS Boot Sector on drive C.

```
_591            3196 8E00 7512 B002 8D1E 9000 B901 00BA 0000 CD26 9DE9 F700
```

Achey_Breaky — **EN:** A simple, appending, direct infector about 836 bytes long. The last byte of every infected file is 0D9h. The virus contains a plain text message: ‘Hi folks, I’m the Billy Ray Virus and I’m here to sing my number one... smash hit, Achey Breaky Hard Disk’.

```
Achey_Breaky    B43F B901 00BA 4404 81C2 8000 CC89 D680 3CD9 7503 E9C9 00B8
```

Algerian — **ER:** An encrypted, 1400-byte appending virus, containing the text: ‘The Algerian Virus Version 1.00, 1993 by Ahmet Cezayirli u Istanbul University Electronics Engineering’.

```
Algerian        7416 B954 0590 8D3E 2401 2E8B 3602 012E 313D 2E31 3547 E2F7
```

Antibase.1900 — **CER:** An encrypted, prepending, 1900-byte virus from Bulgaria. Messages include: ‘AntiDBASE by Michael’, ‘This program is written in the city of Sofia by MIchael’, ‘HEY! STUPID GET AWAY’.

```
Antibase.1900   8C16 6008 0E17 BC09 012E A108 085B 33D8 535B 81FC 0708 72F5
```

AntiGUS — **CR:** A polymorphic, appending virus about 1570 bytes long, armoured with anti-debugging tricks. The decrypting procedure has a constant length with randomly spread ‘junk’ instructions. If we consider the ‘Are you there?’ call, the virus should probably be named ‘Bad Beef’ (if AX=BEEF, Int 21h returns the values AX=0BAD, DX=BEEF). The pattern detects it in memory.

```
AntiGUS         4B4C 2C4D 616C 6179 7369 612E 003D EFBE 740A 80FC 4B74 19EA
```

Baba — **CR:** An appending, 356-byte virus named after its ‘Are you there?’ call: AX=BABA, Int 21h returns the value AX=FACC.

```
Baba.356        BF00 0181 C649 01B9 0400 FCF3 A45E B8BA BACD 213D CCFA 7503
```

CaNTaNDo — **CN:** An encrypted, appending, 857-byte direct infector containing the text: ‘CaN_TaN_Do_v01 : “Onkos tll kilttej lapsia?-)” ’.

```
CaNTaNDo        E8EF FF8B FDBE B931 038A 0430 05D2 0530 0D47 46E2 F4E9 7409
```

Cavaco — **CER:** An appending, 1470-byte virus related to the Sarampo virus [*see VB July 1995 p.5*] - it has a similar payload, and triggers on the same days. The virus contains the text: ‘Do you like this Screen Saver? Cavaco - A virus created by The Portuguese Government’.

```
Cavaco          BAAE 03B8 2125 CD21 B42A CD21 81FA 1904 740F 81FA 190C 7409
```

Deliver.Digi — **CER:** A new, polymorphic, appending, 3547-byte virus from Eastern Europe. The template below can identify the virus when active in memory.

```
Deliver.Digi    C606 EA0D 802E A3EC 0D80 F478 80FC 6974 5ADC C480 FCD4 7453
```

Fasolo — **CN:** An overwriting, 176-byte virus containing the plain-text message: ‘Fasolo VIRUS’. It carries a destructive payload which triggers on December 4th, overwriting the first 128 sectors of the first hard disk and corrupting the CMOS setup data.

```
Fasolo          B403 B080 B500 B101 B600 B280 BB00 00CD 13FB B010 E670 B000
```

Green Monster — **CR:** An appending virus, about 711 bytes long, which contains this plain-text message: ‘Green Monster. I”m happy’.

```
Green Monster   B890 35CD 2107 81FB 3412 7506 1FB8 0001 50C3 8CC8 488E C026
```

HDZZ.566 — **CR:** An appending, 566-byte long virus containing the texts: ‘HDZZ’, ‘COMSPEC’, ‘ITS’.

```
HDZZ.566        9C33 C08E C026 813E F603 B607 7503 EB66 9026 C706 F603 B607
```

Jerusalem.CVEX

CER: This month sees a whole set of Jerusalem-based viruses, from the *Computer Virus Extended Association* (*CVEX*). Each variant is 5120 bytes long.

```
Jerusalem.CVEX.A    FCB8 ACAC CD21 3D30 AC75 10B8 ECAC 2E8B 0E0A 01BF 0001 BE00
Jerusalem.CVEX.B    FCB8 50AC CD21 3DFF AC75 10B8 50EC 2E8B 0E0A 01BF 0001 BE00
Jerusalem.CVEX.C    FCB8 60AC CD21 3DFF AC75 10B8 60EC 2E8B 0E0A 01BF 0001 BE00
Jerusalem.CVEX.D    B861 AC9C 2EFF 1E20 013D FFAC 7514 B861 EC2E 8B0E 0701 BF00
Jerusalem.CVEX.E    B862 AC9C 2EFF 1E20 013D FFAC 7514 B862 EC2E 8B0E 0701 BF00
Jerusalem.CVEX.F    B863 AC9C 2EFF 1E20 013D FFAC 7514 B863 EC2E 8B0E 0701 BF00
Jerusalem.CVEX.G    B870 AC9C 2EFF 1E1C 013D FFAC 7514 B870 EC2E 8B0E 0701 BF00
Jerusalem.CVEX.H    B870 AC9C 2EFF 1E18 013D FFAC 7514 B870 EC2E 8B0E 0701 BF00
```

Jerusalem.CVEX.6140

CER: A polymorphic variant of the Jerusalem.CVEX family for which no simple pattern is possible.

Jerusalem.CVEX.6400

CER: Another polymorphic variant of Jerusalem.CVEX. Again, no simple pattern is possible.

Job.405

EN: An appending, 405-byte, direct infector which will hit only one file at a time. It contains the text: 'I did my job :)'.

```
Job.405             E800 005D 81ED 0301 83EC 5690 8BFC 061E 0E1F 0E07 578D BE6A
```

Jungle

CR: An encrypted, appending, 1000-byte virus. It contains a short rhyme of questionable taste which begins: 'I walked in the jungle…'. Apart from infecting files on execution, the virus targets and infects two particular programs, C:\DOS\MODE.COM and C:\DOS\KEYB.COM.

```
Jungle              8DB6 0401 2E8A 8603 01B9 9C03 2E30 0446 E2FA C38C D08E D88B
```

Lyceum.737

CER: Another appending variant, this one 737 bytes long, extending the Lyceum group of parasitic viruses from Russia.

```
Lyceum.737          E800 005E FC50 53B8 D0AF CD21 3DFD 0A74 7B56 1E06 8CC0 48B9
```

Mickie

CER: An encrypted, appending, 1100-byte virus containing the texts: 'Mickie Lives...somewhere in time!', '(c) 1995 Grave Lion'.

```
Mickie              2E81 3E00 00CD 2075 0803 0607 018E D8EB 028E D8B9 1401 BE31
```

Pizelun

CER: An encrypted, appending 3599-byte virus containing the text: 'PIZELUN attivato, attivatissimo! Premere un tasto per continuare'.

```
Pizelun             E930 00B8 ???? BF?? ??01 F7BA F40D 01F2 3105 47D1 C039 D775
```

Plague

CER: A 2647-byte appending virus with stealth capabilities. It contains the text: 'PLAGUE'. Plague installs itself in memory, and hooks Int 21h, but does not modify the contents of the interrupt vector table.

```
Plague              E819 0006 1E0E 1F8C C82B 061A 018E C0BE 3D0B BF00 01FC A5A4
```

Right2Life

CR: An appending, encrypted, 805-byte virus. When active in memory, it does not allow deletion of any files, and displays the message: 'I'm sorry, but I can't let you mercilessly slaughter poor innocent files in their prime! Files have a RIGHT TO LIFE too'. Because of the specific file date/time stamp, listing of infected files produced by the 'DIR' command shows only the names of files.

```
Right2Life          BD?? ??FC EB01 908D B61A 008B FEB9 0503 8AA6 1F03 AC32 C4AA E2FA
```

RMC.1551

CER: An appending, 1551-byte virus containing a payload which triggers on February 22nd. On this date, large, flashing text appears at the top of the screen: 'HAPPY BIRTHDAY MIRCEA!'. The second message, of normal size, appears underneath and begins: 'THIS VIRUS WAS MADE BY RACASAN MIRCEA AND TODAY IS HIS BIRTHDAY....'.

```
RMC.1551            FF2E 3B05 B404 CD1A 81FA 1602 750C B0AD E664 E825 00E8 0300
```

Rogue

CER: An appending virus which contains the text: 'DBF CHKLIST ??? Now you got a real virus! I'm the ROGUE'. In the 1208-byte variant, the message is encrypted.

```
Rogue.1208          EA00 00FF FF06 1F2E 833D 0174 03E8 AB00 B8B6 032B F82E 0300
Rogue.1213          EA00 00FF FF06 1F2E 833D 0174 03E8 AB00 B8BB 032B F82E 0300
```

SillyRC.403

CR: A simple, appending, 403-byte virus.

```
SillyRC.403         BA8A 00B8 2125 CD21 1F58 508E C0BF 0001 57BF 8301 B90C 00F3
```

Stardot.900

CEN: An appending, 900-byte, direct infector, related to the Stardot virus.

```
Stardot.900         C6FF A16C 0426 A3CD 041F 06B8 2435 CDFE 891E D304 8C06 D504
```

SviSyk

CR: An appending, 252-byte virus containing the plain-text message: '< SVINOLOBOVA SYKA! >'.

```
SviSyk              7413 F3A4 BE84 008E D9A5 A5C7 44FC 5500 C744 FE60 001F 07C3
```

VInfo.666

CER: An appending, 666-byte virus containing the encrypted text: 'VIRUS INFO'.

```
VInfo.666           E800 005B 83EB 032E 807F 775a 7410 BF00 01BE 7600 03F3 57A5
```

Virogen.1520

CER: A polymorphic, appending, 1520-byte virus containing the encrypted text: '(c) 1993 Virogen ASeXual Virus v1.00'. The pattern identifies the virus in memory.

```
Virogen.1520        80FC 0E74 1B3D 004B 7505 80FD FB75 1117 5C07 1F5D 5E5F 5A59
```

INSIGHT

Principles of Detection

Pavel Baudiš belongs to the serried ranks of the unknown, but for how much longer, remains to be seen. Producer, developer, and vendor of a highly-successful anti-virus product, *AVAST!*, the world beckons to him from his home in the Czech Republic, and his offices at *Alwil Software*.

Of course, it was not always like this - in fact, unlike so many people involved in this business, computers played virtually no part in his life until he was an adult: 'Naturally, I had been fascinated by computers since I was a small child,' he recounted. 'But twenty years ago, there were not very many computers in my country.'

Where it All Began

His childhood presaged nothing of what was to become his life's work. In fact, it is almost surprising that he chose the direction he has: 'My parents have nothing to do with computers at all - my father is a psychiatrist, my mother is a child doctor.'

Baudiš read mathematics and engineering at university, studying computers in chemical technology. His main interest was in areas of computer science not often associated with anti-virus software developers: cybernetics and process control. He did not, however, apply his knowledge practically, and when he took up a position at the *Research Institute of Mathematical Machines*, after completing his studies, he started to deal with computer graphics.

'I started with some programmable calculators and then with DEC's PDP clone. Later I worked on HP minicomputer clones. I wrote a PDP program to control an analog computer. On HP I was a member of a team which implemented a version of GKS (Graphical Kernel System - an attempt to standardize computer graphics).'

Unsurprisingly, much has changed in Baudiš' life since he took up that post: 'They mostly use mainframes, and other computers,' he reminisced. 'It used to be one of the best research institutes. I worked with computer graphics, and implementations on mini-computers.

'Everything has changed since 1989, since Vaclav Havel's Velvet Revolution [*so called because of Havel's devotion to the music of the 60s cult band, the Velvet Underground. Ed*]. Since then, this has become a very interesting place - very amazing changes.'

'Much has happened during the last five years. I'm still fascinated, not only by these changes, but also by the speed with which they've happened. It is curious to see also that many people have very quickly forgotten how it was here before… Now I have the chance to run my company, to travel where I wish, to meet interesting people, to be a "normal" European…'

Not the least of these changes was the opening-up of trade routes to the West: doubtless this has helped give Baudiš a higher profile. It would have been unlikely, in the days of Communist rule, for this man even to have participated in an interview, let alone to distribute his product freely in non-Eastern Bloc countries.

The Root of the Matter

AVAST! was conceived in response to his first encounter with a computer virus - that old favourite, Vienna: 'It was spring,' he recalled. 'I played with the virus for a few days, and then began, as many other people, to develop an anti-virus program. It was another year before I met a second virus, Cascade.1701. At the same time, I became aware of the Dark_Avenger virus.

'All this time, I was thinking about how to deal with viruses,' he continued. 'It's very interesting how we started to develop general anti-virus programs. For example, our checksummer was created when we discovered Vienna, or its foundations laid - at the time, there was only one virus.'

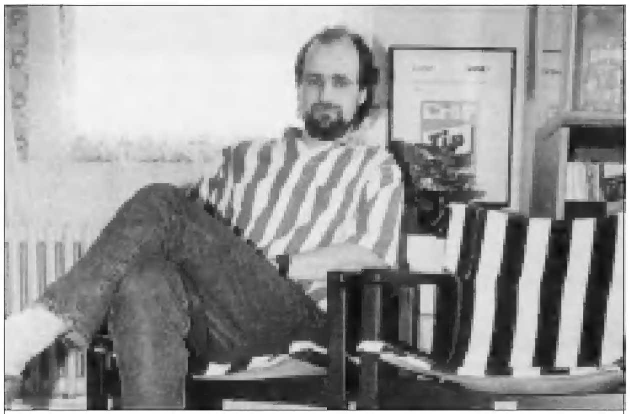
In 1988, Baudiš and his colleagues concentrated on general anti-virus programs - checksum programs, behaviour blockers; programs which do not change much over time. *Alwil Software* was established in 1991, set up to market these products which had, at least in part, already been written. Baudiš and his partner, Eduard Kucera, decided not to focus exclusively on anti-virus software, but also on general data security: 'Really, the anti-virus field is just part of the more complex issue of computer security.'

The company has another flagship product, *Sup* - 'Czech for "vulture",' laughed Baudiš. 'It is a general security program; access control with disk encryption and disk decryption programs. *Sup* is widely used in the Czech Republic.'

AVAST! is a huge success in its native land: the product is in place on some 60,000 machines in the Czech Republic and in Slovakia. Site licences for government institutions, banks and army abound. Trusting its success on home soil, *Alwil* now targets a wider market: they have been represented at *CeBIT* in Hanover three times already, and regularly send delegates to international security and virus conferences.

Company Care

Alwil Software itself does not distribute, even in the Czech Republic - Baudiš has taken the unusual step of setting up another company, called *Alwil Trade*, which is responsible for the commercial side of the enterprise.


Pavel Baudiš has helped to develop one of the most efficient polymorphic virus detectors available today.

'*Alwil Software*,' said Baudiš, 'is just for development and technical support, and everything related to it. We now have about six programmers working on all these products. The anti-virus software uses much more of our resources than the access control product. So, currently there are five people working on the anti-virus technology - one with the Windows environment and Windows programs; two with the virus laboratory, and so on. I still have overall control here, because I used to develop all the parts, so I want to see how it looks.'

Alwil Trade employs approximately ten people: it is not involved exclusively in distributing the program produced by its sister company, but also sells other products. Baudiš is now trying to penetrate the international market, but does not find it easy.

'I think that the market for anti-virus programs is tied up in some kind of price war,' he said, 'so it's sometimes very difficult to try and get into this market. However, I think that there is a way - we are trying to establish distribution channels, especially in Europe. It's a slow process, but I think it works, and things like the *Virus Bulletin* tests and reviews help very much, because they bring our product to people's attention. Independent testing helps very much.

'We are not looking for "box movers" to distribute for us; we want companies which are competent and able to do what we require. We really have a very high level of support in the Czech Republic for all of our users, so we wish to establish similar support in other countries too. It is my conviction that the relationship between the customer and the anti-virus company starts from the moment he buys a particular product.'

Baudiš is pleased with the fact that *AVAST!* is now available in several European countries: 'Our first target is Europe, because of its proximity to the Czech Republic. The American market has special problems. I have seen one company which spent a lot of money in the US, and built its success on the American market. However, our first aim is to establish ourselves here in Europe - we will decide where to go then on the basis of what happens there.'

The Other Side of the Fence

He takes a traditionally ambivalent stance towards virus writers: 'Many virus writers are just kids playing a dangerous game. On the other hand, many know very well what they are doing and that they can cause much trouble to other people. Such people can be viewed as criminals.'

Legislation in the Czech Republic, he feels, is in quite a good state - there are laws which can be found to apply to virtually all sorts of computer crime: 'There is not a special computer law, however, and of course no "anti-virus" law at all. The main problem here is similar to many other countries - to prove to the virus author that he/she wrote a particular virus, and to calculate what and how much damage this virus caused. Also to prove that the virus has been written to cause this damage.'

Viruses, he feels, are here to stay: 'I'm afraid there is no chance at all of stopping people writing viruses. You can always find somebody who has technical knowledge exceeding moral responsibility. It is very difficult to explain to him that he is doing something wrong.'

Looking Ahead

In his view, the recent trend towards polymorphic viruses will continue. 'I think we will see more of these, and more viruses which are able to escape heuristic scanning. Unfortunately, I can also see viruses being targeted against new operating systems such as *Windows NT* and *Windows 95*.'

To cope with these problems, Baudiš sees anti-virus software of the future as a mixture of different techniques, based on scanning, checksumming and behaviour blockers. The network-oriented part, he believes, will be of much higher importance: 'As viruses will spread over networks (and not via floppies, as they do today), the need for fast detection, report and reaction will be of greater importance. I think this direction will be followed by all anti-virus software producers - not only here in the Czech Republic.'

The Home Front

Baudiš, at 35, has been married for longer than he has been involved with viruses: his wife is a teacher in a special school for blind children and professes to hate computers ('I have no idea why,' he chuckled).

The couple has two daughters, aged six and eight. Although there is a computer in the family home, the girls play with it only occasionally, taking it as another part of the everyday. Baudiš is concerned that they keep a well-balanced outlook on life: 'There are other things which are important to them - they play musical instruments (piano and flute), read books, draw very nice pictures… Simply said, they will have enough time for computers in the future.'

A pragmatic outlook from a pragmatic man - and this cool and collected attitude will no doubt stand him in good stead in the very intense and fast-moving world of viruses.

VIRUS ANALYSIS 1

DieHard2: Hard and Alive!

Eugene Kaspersky
KAMI Associates

It is the great numbers of new viruses passing through the hands of anti-virus researchers which account for the fact that some complex viruses are not analysed completely. The only information needed from such viruses is detection and disinfection methods and trigger routines - other features are not discussed. This is a pity, but there is simply not enough time to analyse each virus down to its last byte.

Sometimes, however, it is necessary to return to such viruses, and look more closely at their code. DieHard2, which first appeared about a year ago, is one. At that time, it was analysed, and included in the anti-virus databases of dozens of scanners - but is still 'In the Wild'. This is a good reason to unpack and scrutinise dusty files with this listing.

First Impressions

The virus is large (files increase by 4000 bytes on infection), encrypted, and uses techniques which target debuggers. This indicates that analysis will be difficult and time-consuming.

First, the virus must be decrypted. The decryption routine is not polymorphic, nor hidden in any other way. It receives control directly in COM files - in EXE files, the entry point is the address of this routine. The code of the decryption routine is quite specific, and may be used as the scan pattern to detect an infected file.

This is not the most complex part of the analysis: problems appear at the same time you try to debug the code which decrypts the virus, and start to analyse the virus' installation and infection procedures.

The virus uses a most effective way to stop debugging; redirecting the stack pointer at the bottom of the infection routine. As a result, under the debugger, the decryption routine is destroyed by the stack. It would be destroyed by any hardware interrupt performed during decryption, but the virus disables them with a CLI instruction, saving its own code. Attempts to fool the virus by bypassing the instruction which redirects the stack results in the virus not working after decryption. The stack must be where the virus sets it.

Installation

After some work and emulation with the stack in mind, the virus appears in a non-encrypted form, and the decryption routine returns control to the installation code. This brings with it more surprises and programming tricks. The virus traces a number of interrupts (10h, 13h, 21h and 40h), storing the original addresses to bypass anti-virus monitors.

The virus checks the code being traced for PUSHF/POPF (push/pop flags) and RETF (return far) instructions, and does not allow tracing to be disabled in this manner. While tracing, the virus also hooks Int 08h to allow it to re-enable tracing if a program has disabled it in any way.

It also checks the conditions which cause the termination of the tracing, comparing the code where tracing has finished with several instructions (such as interrupt calls and far jumps). DieHard2 uses a wide range of techniques to find the original addresses of the interrupt, and breaks tracing another time, trashing the Int 03h (BreakPoint) address.

Using information from tracing, the virus checks to see if it is resident, looking for an Int 21h handler with its ID stamp in the segment address of Int 21h XORed with D1A5h. Its ID word varies by machine and configuration, remaining constant on the same PC in the same environment.

Then the virus checks to see if *DESQView* or *Windows* are in use, by calling first Int 21h with values AX=2B01h, CX=4445h ('DE'), DX=5351h ('SQ'), and then Int 2Fh with values AX=1600h and AX=1700h. It does not install itself into memory if *DESQView* or *Windows* are active.

If a virus copy is not resident in system memory, and neither *DESQView* nor *Windows* are running, DieHard2 proceeds to install itself. It uses standard Memory Control Block manipulation techniques (it creates a new MCB, setting its owner to DOS), and hooks Int 10h and Int 21h.

The method by which the interrupts are hooked is unusual - the virus does not store the new interrupt address in the Interrupt Vector Table, but 'inserts' itself into the chain of programs hooking the interrupts. The virus searches for the command passing interrupt flow from one program to another, overwriting that address with its own (see Fig 1).

While making the TSR copy, the virus divides its code into three parts: stealth and infection routine, installation routine, and interrupt handlers. The virus leaves only one of these

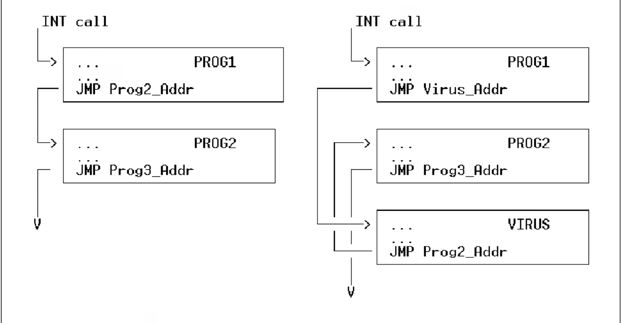


Figure 1: The left-hand diagram shows the Int 21h flow before infection; the right-hand, post-infection Int 21h flow.

parts, the interrupt handlers, non-encrypted in memory. Other parts are encrypted: the virus decrypts them where necessary, executes them, and encrypts them again.

After installation, the virus increases its internal counter: the word containing the encryption key. This is done in memory and in the host file. The virus reads the code containing the counter, increases it, and writes it back to the host. Installation is now complete, and control returns to the host file.

Int 21h Hooking

Int 21h functions intercepted by the virus are: Create (3Ch), Open (3Dh), Close (3Eh), Read (3Fh), Write (40h), Lseek End (4202), Exec or Load (4B00h, 4B01h, 4B03h), Rename (17h, 56h), and Extended Open (6Ch). Hooking them allows the virus to utilise its stealth and infection algorithms.

The stealth algorithm is used on file access. It does not allow the body of the virus to be seen, and substitutes the file header with the original. Only one stealth routine is missing from the virus code - the subtraction of the virus length from infected file length during FindFirst/Next calls.

The virus has a ‘good’ length (exactly 4000 bytes) and the increased length of infected files does not attract attention. Moreover, viruses which use stealth on FindFirst/Next calls have to intercept the execution of CHKDSK to prevent error messages, but DieHard2 does not have to do this.

On execution, opening, closing, or renaming of a file, the virus calls the infection routine, which checks for the virus’ ID word, A5D1h, at offset 60h from the file’s entry point. In COM files, the entry point is calculated according to the JMP instruction (if present) at the start of the file header. If the file is not infected, the virus reads, encrypts and stores its header, encrypts and saves the virus body at the end of the file, and overwrites the file header with a JMP Virus instruction (3 bytes) in COM files, or corrects the header of EXE files to pass control to the virus code on execution.

Where necessary, the virus checks the file name for COM and EXE extensions. It does not infect IBMDOS.COM or IOSYS.COM. During infection, the virus hooks the Int 24h handler to prevent error messages while attempting to write to write-protected disks, and temporarily hooks the Int 13h and Int 40h handlers to disable the anti-virus monitor alarm.

So, DieHard2 is a real stealth and fast infector. Scanning for the virus in files without memory detection is useless: the scanner will find nothing, and in the course of scanning, the system will become 100% infected.

It is time to say more about the programming style of DieHard2. Everything about its algorithm is easy, but the assembler code is crazy, and difficult to understand. There are sequences of subroutine calls which do not return to the call point, a number of switches, dozens of data buffers and flags, etc. In short, it took me two hours to discover how the virus recognizes infected files, and two hours to find the ID word and the offset where it is placed (I’m a diehard too :-)).

Trigger Routines

The virus has several trigger routines. On any Tuesday which is also the 3rd, 11th, 15th, or 28th day of the month, the virus may, on calls to the DOS function Write, write the following message to the Standard Error screen and AUX devices (usually COM1):

```
SW Error
```

The second trigger routine writes strings into PAS and ASM source files. These strings are code for programs which have identical effects: the compiled programs display two bytes on screen, displayed as Chinese characters; D1h and A5h. Incidentally, these bytes are the same as the ID word of the virus - A5D1h.

The virus also uses its stealth algorithm with corrupted ASM and PAS files. Under infected system conditions, this source code is not visible in ASM and PAS files.

The last trigger routine is performed from the Int 10h handler. The virus checks its generation counter, and, if equal to or greater than 15, and if the current video mode is number 13h (graphic), the virus displays ‘SW’ in large violet symbols. DieHard2 also contains the internal text string ‘SW DIE HARD 2’: this gives the virus its name.

DieHard2

Aliases:	DH2, SW, Die_Hard, Die_Hard.4000.
Type:	Memory-resident, encrypted, parasitic file infector with stealth capabilities.
Infection:	COM and EXE files.

Self-recognition in Files:

The word at offset 60h from file entry point is compared with the ID word A5D1h.

Self-recognition in Memory:

Int 21h handler data checked with ID word, which is variable according to environment.

Hex Pattern in Files and in Memory:

```
E800 005B 8D7F 5B0E 07FD AB8B
C3B1 04D3 E840 8CCA 03C2 8BD0
```

Intercepts:	Int 21h (permanently) for infection and Int 10h for trigger routine, Int 08h during interrupt tracing, and Ints 13h, 24h and 40h during infection.
Trigger:	Overwrites ASM and PAS files, displays messages.
Removal:	Under clean system conditions, identify and replace infected files.

VIRUS ANALYSIS 2

Crazy_Nine: Paving the Way

Matt Brown

It has never been so easy to discover the inner workings of the PC: Ralf Brown's Interrupt List and books like *Undocumented DOS* and *The Undocumented PC* are readily available, comprehensible and reasonably complete.

The Crazy_Nine virus is a product of this new knowledge: it uses all kinds of low-level or undocumented techniques to evade detection of possible viral behaviour. Indeed, some of its routines appear to be copied almost line-by-line from some of these books.

It is 4 KB long, making it one of the larger boot sector viruses I have seen; partly due to the low-level nature of a lot of its code (doing things oneself is always more work than getting the BIOS to do it) and partly due to its (albeit limited) polymorphism in the boot sector - polymorphism of any kind is rare in boot sector viruses.

Installation and Infection

On floppies, the eight sectors of virus code, plus the original boot sector, are located two tracks beyond what is normally the final track (for example, track 81 on 3.5-inch, 1.44MB diskettes). This has several advantages - it will work on most PCs, the sectors are not likely to be overwritten later (and data is unlikely to be damaged on infection), and the sectors are not easy to find. The polymorphic code loads these sectors in memory.

When this has been done, it branches to offset 0E8Bh in the newly-loaded code. After obtaining the Int 13h vector, it copies the original, uninfected MBS code to memory location 0000:7C00h, over its own MBS. An 'Are you there?' call is then made; Int 13h, AX=5445h. If the virus is resident, it simply returns. Most boot sector viruses do not need such a call, but Crazy_Nine may run twice: the virus turns off floppy boot, and after running from the MBS, will run the boot code from the diskette, if present.

Crazy_Nine then loads the MBS and searches through its first 40h bytes for the word 0500h - the decryption instruction (an ADD) in the polymorphic MBS code produced by the virus. This can be used as a quick and dirty detector for the virus; standard DOS MBSs contain no such value.

If the MBS is not infected, the virus writes itself to the last nine sectors of the cylinder *above* that which the BIOS reports (via Int 13h AH=08h) as the last cylinder. This is the diagnostic cylinder the BIOS reserves for testing purposes. It is seldom used, and provides a good hiding place for a virus - in fact, I am surprised not to see such a technique used more often. It is a subtler method than using sectors on the first track; less obvious, and not viewable by many disk editors. Most disk editors perform sanity checks on the numbers they are given, and refuse to let the user see parts of the disk outside the area the BIOS reports as valid.

Crazy_Nine then obtains a random number seed by reading the real time clock, and uses this to build a polymorphic MBS in memory. The polymorphism consists of short, two-byte sequences of junk interspersed with fixed instructions. The loop forms a decryptor for the code which loads the body of the virus from the diagnostic cylinder.

Next, the virus calls a routine which writes the MBS to disk. It reads byte 12 from the CMOS, which indicates the hard drive type: if this states that there are no hard disks, it usually means that the drives are on a separate, probably non-AT compatible, adapter card (e.g. SCSI). If that is the case, the virus will abandon any attempt at clever tricks, using the standard Int 13h interface to write to the MBS.

"Crazy_Nine uses many tricks which will probably become much more common"

If the drive appears to be AT-compatible, a more cunning technique is used: the hard drive controller is driven using standard AT port commands. This enables the virus to go underneath most behaviour blockers, which usually operate at BIOS level. With the proliferation of such behaviour blockers, this is a neat trick.

Having infected the MBS, the virus disables the floppy drives in the CMOS in a manner similar to EXE_Bug. This prevents floppy booting under certain BIOSs, but does not make drives inaccessible from DOS: if this is attempted, the system boots from the hard disk, and runs the virus.

Crazy_Nine then checks to see if there is a diskette in the drive: if there is, it runs the boot code from the floppy. Thus, the user does not become suspicious, in spite of the fact that the virus has still been run.

The virus then creates (at FFFF:0200) a JMP FAR instruction to the original BIOS Int 13h entry point. Since the A20 gate is in its default state immediately after boot (which means that some of every segment above segment F000 will wrap around to the base of memory), this memory reference is equivalent to 0000:01F0. This location is in the Interrupt Vector Table - it overwrites the vectors for interrupts 7Ch and 7Dh, which are seldom used. Crazy_Nine then sets the Int 13h vector to point to FFFF:0200, to foil any behaviour blocker which tests the interrupt vectors to see if they point to the BIOS. Most just test to see if the segment is greater than F000h, assuming that only the BIOS is above it.

Next, the virus saves the interrupt vectors of Int 1Ch and 21h, and the value at 0000:0700 (the location at which IO.SYS is loaded). The virus then intercepts Int 1Ch.

Resident Behaviour

Crazy_Nine intercepts Int 1Ch to find out when DOS has been loaded, and hooks interrupts after this has happened. After the Int 21h vector has been changed three times (the number of times the vector is changed during DOS loading in recent versions), the virus overwrites the first five bytes of the Int 13h handler with a JMP FAR to its own code, saving the original bytes to be restored when it calls the original Int 13h handler. Thus, the Int 13h vector still points where it should, not at the virus code, making things seem less suspicious to programs attempting to monitor viral activity.

Next, the virus calls a routine which uses the 8259 Programmable Interrupt Controller (PIC) to change which processor interrupts are generated by the hardware IRQs. Normally the IRQs from the first PIC generate interrupts 08h-0Dh; Crazy_Nine reprograms the PIC so that a different range of interrupts, higher and normally unused, is generated instead.

The virus installs interrupt handlers for these new IRQs. With the exception of IRQ 0 (normally Int 08h, the timer interrupt), they just call the original handlers for these interrupts. The IRQ 0 handler tests that the current Int 13h handler contains as its first instruction a call to the virus. If not, it installs one, to ensure that the virus is called first on the Int 13h chain. This allows the virus to ensure continually that it has not been deactivated in memory.

As IRQ 0 no longer generates Int 08h directly, any anti-virus software which checks to see if Int 08h is behaving as it should will be fooled. I know of no software which checks PICs to see what interrupts they are really generating.

Reads or writes to hard disk MBSs are stealthed; however, no such action is attempted for floppy disks. Also intercepted are Int 21h functions Read, Write, Verify and Detect Disk Status (AH = 02h, 03h, 04h, and 16h) for floppy drives; these will attempt to infect the floppy disk.

Infection

The floppy boot sectors created by the virus are somewhat polymorphic, but do not contain a decryptor; instead, they simply insert junk between the useful instructions. The virus builds such a boot sector in memory, and then writes it to the disk.

As in the hard disk infection described above, the virus avoids the Int 13h interface where possible; instead, it programs the floppy controller and the DMA controller directly. This is only done if the processor is in real mode, and not V86 mode (as it is under a memory manager or within a *Windows* DOS box). The virus tests this by executing a SMSW AX instruction and testing the PE (Protection Enabled) bit of the returned status word.

Within V86 mode, use of I/O ports can trap to the V86 monitor, so the program can no longer rely on gaining stealth by using them. Additionally, some V86 environments (*Windows* DOS boxes, for example) do not emulate all I/O ports, since well-behaved software will not use them.

Trigger and Dropper

Crazy_Nine does not appear to contain any trigger routine: this is unsurprising, given the size of its installation and infection routines - at 8 sectors (4 KB) long, it is large for an exclusively boot-sector virus. It should nonetheless be noted that infected machines have been known to crash frequently.

I obtained droppers for the virus within a PKZIP archive: PKZIP.EXE and PKUNZIP.EXE were both droppers. Both functioned correctly, so the user's suspicions would not be immediately aroused, unlike the PKZ300B Trojan recently discovered. The dropper is complex, with many anti-debugging tricks, but as it is not necessary to trace its execution to notice it dropping the virus, it is a bit pointless.

Conclusions

Crazy_Nine uses many tricks which will probably become much more common, now that they are better documented. In particular, port-level infection on floppy and AT-compatible hard disks is not too difficult, and the use of the diagnostic cylinder on hard disks is a simple trick.

The increased documentation on low-level aspects of DOS and the PC is a double-edged sword - whilst it makes viruses such as Crazy_Nine possible, it also simplifies the job of the anti-virus researcher. It is the latter for which I am grateful.

Crazy_Nine

Aliases:	None known.
Type:	Boot sector and MBS; resident; stealth.
Infection:	DOS boot sectors of floppies, MBS of hard disks.

Hex Pattern in Floppy Disk Boot Sector:

```
BB * 8EC3 * 8BD8 * B8 * B9 *
B809 02 * BA * CD13
```

Hex Pattern in Hard Disk Boot Sector:

```
8ED8 * B0 * BF * 0005 * 47 *
FEC0 * 75
```

Hex Pattern in Memory:

```
9C3D 4554 74CD E8FC 0C81 FA80
0075 1383 F901 750E 80FC 0377
```

Intercepts:	Int 08h and Int 1Ch for installation; Int 13h for infection and stealth.
Trigger:	None.
Removal:	MBS - FDISK /MBR, floppies - SYS.

VIRUS ANALYSIS 3

Darth_Vader

Jim Bates
President, Institute of Analysts and Programmers

In the nine years or so since PC viruses first appeared, a great deal of hype and hysteria has flourished, then withered. The average computer user is now better informed about viruses, and the problems they can cause. There are still, however, occasional outbursts from anti-virus vendors, and now and then a virus researcher discovers a technique which he is certain is a prelude to Armageddon.

New Topics, Old Themes

The truth is that virus writers are running out of ideas - only so many functions can usefully be subverted. Occasionally, a new idea pops up, but such occasions are becoming rarer. The introduction of greater functionality in operating systems gives rise to new ideas, but the additional capabilities are almost invariably more complex to subvert. Another old chestnut which seems to have disappeared into the woodwork is the 'beneficial virus'. If this beast really did exist, we would probably have heard about it by now.

While I was thinking about these things, I began looking back over old disassemblies, examining them for some of the trickier techniques virus writers have tried. A productive area for such discoveries was among Bulgarian viruses: I was surprised to find a particularly nasty little virus which has not yet been analysed in *VB*.

Becoming Active

The virus in question, Darth_Vader, came to me as assembler source code files and some infected sample programs; all collected from a virus exchange BBS. The assembler files purported to be the author's original source code (modified in one case by someone else) and comprised a series of four versions of the virus, using the same techniques.

This virus uses two main techniques - the first concerns how the code becomes resident and active within an infected system. Instead of the usual method of stealing memory for its own use, Darth_Vader inserts its code into an area within the *MS-DOS* code memory segment.

This is achieved by searching the DOS memory segment for a continuous section of zeros (200 to 250, depending on the version of the virus). The virus copies itself into that area, on the assumption that it will remain unused - this may not work with some versions of *MS-DOS*.

The system services are hooked by a process in which the same DOS segment is searched for a highly specific series of instructions within a routine called the DOS dispatcher. This routine examines commands issued by application software and the higher levels of DOS, then dispatches them to the appropriate system processing routines.

The virus recognises a particular sequence of commands and extracts the target address of one of the dispatch routines. The original address is stored within the virus code, and the dispatch routine altered to ensure that certain file write access commands are dispatched through the virus code.

This has the same effect as the more mundane (and more reliable) hooking of interrupt vectors, and has a slight edge in that it may be more difficult to detect and recognise.

Infection

Once installed, the virus is able to monitor the system requests which pass through it, subverting them where necessary. At this depth within the operating system, normal system functions cannot be implemented; thus the virus uses undocumented internal functions to achieve its aims.

The file write functions are intercepted before they reach the operating system, and operations are conducted on the contents of the write buffer. Darth_Vader does not contain a write command: this could cause problems for certain types of anti-virus software. The virus uses the write functions to determine whether the target file has the extension '?OM', and if the file pointer is set to the beginning of the file.

If these criteria are met, the virus searches the write buffer for a block of zeros, in a similar manner to that used during installation. If found, the virus code is copied into it, and the first three bytes of the buffer change to a jump instruction: this ensures that the virus code will be executed. The original three bytes are stored within the virus code, so an infected file may be repaired prior to proper execution.

Once all this is done, the dispatch routine is sent on its way with the modified write buffer. It is the system which actually completes infection, by writing the file contents.

'Could Try Harder'

The idea behind installation and infection seems to be that the virus code should co-exist as closely as possible with DOS (physically and logically), making it difficult to detect and identify. In fact, the virus presents no problem even to the simplest scanning software, but there are other considerations. These concern the fact that the virus is written during normal system activity. During infection the system is writing a COM file - this has interesting ramifications.

A first-line anti-virus defence mechanism is the integrity checker, which checks potential infection targets against known details for changes. This method depends for success on the software being correctly 'introduced' to the files it must protect, so that relevant information may be collected and stored for future reference.

As a COM file written by the system must be a new one, it will not have passed through this introduction, and can thus be expected to avoid the attentions of integrity checking software. At first, this does not seem to present much of a threat, as COM files are program image files: most computers in ordinary use only execute them; they do not create them. Or do they?

Consider what happens when you copy a COM file - the system creates a new COM file on the target drive and copies the contents of the source file there. Then, the date and time are set to match the original, and the process is completed. So, if a COM file is copied on an infected PC, the original (which may be protected by an integrity checker) remains uninfected; the new copy carries the virus.

This process may not happen often, which is why the method is called *slow* infection. It can be effective, however, and highlights a weakness in integrity checking software.

Conclusions

The techniques described here have been in existence for some time, and do not represent a serious threat for anti-virus software. The techniques displayed within the various versions of Darth_Vader are unreliable, and are limited to certain DOS versions. The principle behind it is well-known within the virus writing community.

Darth_Vader (I, II, III, IV)

Aliases:	None known.
Type:	Resident, parasitic, inserting slow infector.
Infection:	COM files containing at least 202 consecutive zeros.
Self-recognition in File:	Non-encrypting, so recognises its own code.
Self-recognition in System:	Redirects the internal DOS command dispatch routine.

Hex Pattern:	`26AD 3D2E 8B75 F826 AC3C 7574`
	`233C 9F75 EE26 8B34 B9CA 008D`

Intercepts:	Dispatcher calls.
Trigger:	None.
Removal:	Locate and replace infected files under clean system conditions.

TUTORIAL

More Bits for Your Buck

One of the most beneficial features of *WfW* (*Windows for Workgroups*) *v3.11* is 32-bit file access, which, curiously, is almost ignored by the user community. However, with the imminent advent of *Windows 95*, 32-bit file access will move into the mainstream, and become significantly more of a problem for memory-resident anti-virus software.

'Don't I have Enough Bits Already?'

Why might people use 32-bit file access? Well, the system was designed to speed up *WfW*, presumably in recognition of the fact that things were becoming a little slow on most hardware, due to the steadily increasing size and complexity of the *Windows* system.

To reduce the whole system to a snappy catchphrase, it may be said that 32-bit file access does for Int 21h what 32-bit disk access does for Int 13h.

As described in a previous *VB* article [*Enhancing Your Chances, May 1995, p.18*], 32-bit disk access provides faster low-level access to the disk hardware by preventing Int 13h calls from ever reaching the system BIOS, passing them instead directly to the hard disk controller. This speeds things up, not only because of the inefficiencies of many BIOSs, but also by reducing the number of times the processor has to switch between real and protected modes.

It is the purpose of 32-bit file access to reduce the number of Int 21h calls reaching DOS as much as possible. It helps, as far as speed is concerned, to cut the involvement of DOS in the process of file access, for a variety of reasons.

Firstly, like 32-bit disk access, it enables the processor to spend more of its time in protected mode, less in real mode, and (most importantly) less time switching between the two.

In addition, enabling 32-bit file access allows the use of a more efficient, protected mode; a replacement for the SMARTDRV program almost universally used on DOS-based systems. This replacement is VCACHE.386.

Turning on 32-bit File Access

Whilst the concept behind 32-bit file access is reasonably transparent, the location of the dialog is not. Under the Control Panel is an applet (mini application) labelled 'Enhanced'. The dialog revealed by opening this has a button labelled 'Virtual Memory'. Clicking this reveals a window, the top of which (unsurprisingly) describes the virtual memory settings. The bottom of the window shows the status of 32-bit disk and file access. Press the button marked 'Change>>', and the window unfolds. At the bottom are buttons enabling 32-bit disk and file access to be turned on and off (See Figure 1). Exactly why the options are behind a button labelled 'Virtual Memory' is clearly part of the great usability enhancement *Windows* offers.

Tests

The machine on which the tests were done ran *WfW* without a resident scanner. Although 32-bit file access was used as a matter of course, no previous tests had been done on it.

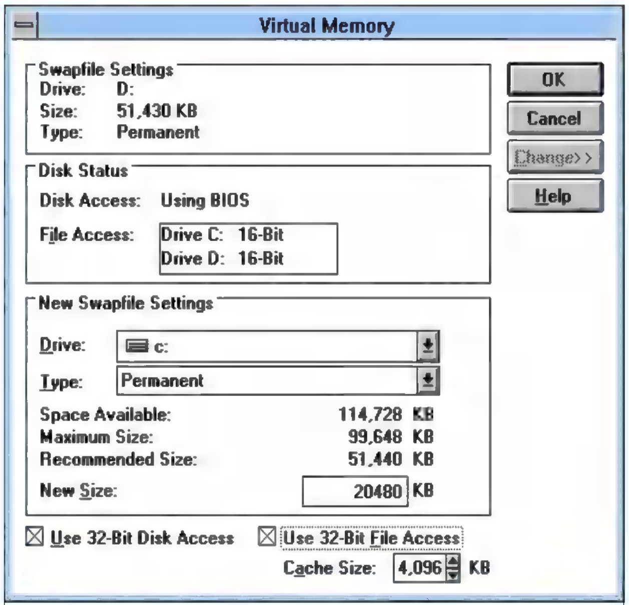
For the tests, INTRSPY was used, which is an interrupt monitoring utility originally provided with '*Undocumented DOS*' by Andrew Schulman.

This resident utility allows easy selection of which interrupts and functions to monitor, and storage of useful information on the arguments for browsing later. A simple INTRSPY script instructing it only to store information about file opens was written, and used to log events as *Windows* started under various configurations.

The Mysterious Results

When *WfW* starts in 32-bit file access mode, a rush of file accesses is usually seen at the start of the process. This is followed by a cessation of all file open activity shortly after VFAT.386 (which essentially contains the majority of 32-bit file access functions) and VCACHE.386 (this, as mentioned above, is the 32-bit file access disk cache module) load.

What I actually saw was no reduction in the amount of file opens as the start-up sequence progressed. Indeed, even when the load process had completed, opening files in text editors resulted in the call going all the way down to INTRSPY and being registered. It appeared that 32-bit file access was not working.



```
Swapfile Settings
Drive:      D:
Size:       51,430 KB
Type:       Permanent

Disk Status
Disk Access:   Using BIOS
File Access:   Drive C:  16-Bit
               Drive D:  16-Bit

New Swapfile Settings
Drive:      c:
Type:       Permanent
Space Available:          114,728  KB
Maximum Size:              99,648  KB
Recommended Size:          51,440  KB
New Size:                   20480  KB

☒ Use 32-Bit Disk Access   ☒ Use 32-Bit File Access
                           Cache Size:  4,096  KB
```

Figure 1: Virtual memory settings and status of 32-bit disk and file access in *Microsoft's Windows for Workgroups v3.11.*

Passing the Protocol: The Multiple Network Problem

The local network to which I connect is not based only around *Microsoft* networking. *NetWare* servers are also present, so it is necessary to load multiple protocols (NetBEUI/NetBIOS and IPX) in order to communicate with all the various servers.

It transpired that, if I prevented all the protocols and helper drivers associated with *NetWare* from loading, and then did the test with INTRSPY, the expected results were obtained. That is to say, INTRSPY ceased to see any file opens shortly after VFAT.386 and VCACHE.386 were loaded.

It seems, for whatever reason, that *NetWare* drivers force all calls being monitored to be passed down to the Int 21h stack, at which point they reach INTRSPY. When the system has determined that the call is destined for a device on which 32-bit file access is supported, it is likely that this call will be sent back up into the protected mode file system: it was not possible to determine this absolutely. There is certainly a speed increase when 32-bit file access is enabled, even if *NetWare* drivers are present.

The Rest of the Story

Once the problem with the *NetWare* drivers had been resolved (by temporarily removing them from the setup), the INTRSPY logs could be examined. As mentioned above, calls to Int 21h, function 3Dh, ceased shortly after the loading of drivers for 32-bit file access - in fact, immediately after KRNL386.EXE (the 'guts' of *Windows*) was loaded.

Contrasting this log with another obtained by disabling 32-bit file access entirely, the impact of 32-bit file access can be seen clearly. Even the size of the log files provides an indication - 19K without 32-bit file access, 3K with.

Anti-virus Products

All this is of significant import to anti-virus products. Clearly, for a simple scanner, there is no problem - the different paths through the interrupt trees are not so important. As long as the scanner receives the files requested, everything is fine.

However, for an anti-virus TSR, the situation is somewhat more problematic. One of the functions of such TSRs is to hook Interrupt 21h, for the purpose of intercepting file activity to check files as they are accessed.

Most of these work by running as soon as possible after the computer boots up - either in CONFIG.SYS, or (more usually) in AUTOEXEC.BAT. It is obvious that this is done in an attempt to gain control of file accesses before a virus is able to get there. The TSR will, of course, check that there are no viruses resident in memory before it loads.

Imagine that a very simple resident anti-virus utility is on the system. This mythical utility simply traps file opens using Int 21h, function 3Dh, and checks the file before the open is permitted. This utility is called *ExeComChecker*[TM] (the reader should note that its design closely matches the configuration of INTRSPY described above).

ExeComChecker loads up, discovers that a virus is in memory, hooks Int 21h, and goes resident. The user then starts *WfW*, having previously enabled 32-bit file access. *WfW* loads, examines the system, and installs 32-bit file access by diverting Int 21h. From this point on, *ExeComChecker* will not see any file accesses.

Even if *ExeComChecker* has followed all the rules of hooking interrupts, and has installed *Windows* drivers in response to the '*Windows* is now loading' interrupt call (Int 2Fh, AX=1605h), it will not see anything further. It is also ironic that it will not even be aware that anything is wrong; the system will simply appear to *ExeComChecker* to be inactive.

When *Windows* exits, *ExeComChecker* will respond to the '*Windows* is now shutting down' interrupt (Int 21h, AX=1606h), and unload its *Windows* drivers. The Int 21h chain will revert to the state in which it was before *Windows* started, and *ExeComChecker* will start to work again.

If the normal operation of *ExeComChecker* is transparent (an unusual occurrence in the field of resident scanners), it is very possible that the user will not notice that the scanner is not trapping anything - he will assume that files are being checked, but coming up clean.

Virtually no Problem

The solution to all this, as anti-virus producers are beginning to realise, involves one of the most confusing acronyms in the *Windows* pantheon - VxD. VxD stands for Virtual Device Driver: this may at first seem a little odd, but the 'x' simply signifies the name of the device being virtualised. It is the job of a VxD to 'pretend' to be a device - where a device driver puts a standard interface onto a physical device, a VxD simulates a device.

So, if the developers of *ExeComChecker* decided that they had to implement a VxD, it might well be called *VExeD*, for 'Virtual *ExeComChecker* Device'. [*Groan. Ed.*]

It is often said that a VxD under *Windows* is the magic solution to a great many problems which *Windows* introduces. But how does it help here?

Using a VxD to hook file access enables the programmer, amongst other things, to intercept file access calls which do not reach the DOS interrupt stack. This is clearly just what needs to be done to trap calls via 32-bit file access which are missed by the standard technique.

How is it Done?

To intercept file access, a VxD must register a hook with the IFS (Installable File System) Manager. Herein lies the problem. For reasons best known to themselves, *Microsoft* has been very coy about giving details on this for *WfW*. However, in the run-up to the release of *Windows 95*, information on how to perform this function on the new operating system has become available. It was found that the system used under *WfW* is almost identical in concept, if not in detail.

There is a service function in the IFS Manager called IFSMgr_InstallFileSystemApiHook(). When a VxD calls this, it passes in a new hook function to call, and is given the next hook down the chain, which must be called by the VxD to link the chain once more.

From then on, when a file system call goes through the IFS (and all should), the hook function in VxD will be called. One argument passed on each call will be an I/O Request (ioreq) structure, which describes the I/O function to be carried out.

Back to *VExeD*

To return to the specifics of a resident anti-virus utility, *VExeD* would be able to ignore all I/O requests other than file opens, and trap them, calling the scanning engine to check the file.

The only problems arise when the VxD itself issues I/O calls, which pass through the IFS Manager and are trapped by its own hook; however, these can be solved with judicious programming.

One advantage which has not yet been mentioned is the fact that using a VxD to check files for viruses enables the code to be 32-bit, giving significant speed and ease of design advantages over 16-bit code.

Conclusions

Until recently, it has been possible for anti-virus software vendors to claim that most customers do not need a VxD. However, within the last six months, ever more computers have come with *Windows for Workgroups* pre-installed: *Microsoft* states that such computers should come with 32-bit disk and file access options configured into the optimal settings for that machine: which presumably means enabled, if they work.

It is to be assumed that an increasing number of manufacturers will want to supplant *WfW* with *Windows 95* as the pre-installed windowing system of choice. To support this, anti-virus software vendors will need to produce resident utilities compatible with the new system, which means they will have to produce a VxD.

The good news is that *Windows 95* has been in beta and pre-production stages for so long now that those companies which have been developing software for the new operating system have had a running start. If *Windows 95* takes off to the extent which *Microsoft* hopes it will (and to an extent to match their marketing budget), the companies which invested the time and effort will be glad that they did.

FEATURE

Viruses in Israel

Shimon Gruper
EliaShim Microcomputers Ltd.

The first virus surfaced in Israel in September 1987. One Friday, articles appeared in all the major Israeli newspapers, announcing that the end of the computer era had arrived.

These voices of doom proclaimed that 'a new disease called COMPUTER VIRUS would destroy all computers and the software in them'. This first much talked about infector was the notorious Jerusalem virus, originally discovered in the *University of Jerusalem.*

At the time this virus was discovered, Israel was a 'one diskette' country. Back in 1987, no one was willing to pay for the use of software, unless it was copy-protected and impossible to duplicate, in which case purchasing it was unavoidable. Users in small and large organizations were illegally copying software, a fact which was considered normal behaviour at the time.

Software houses had a hard time selling their products because people were not able to understand why they should have to pay for them. This led to the introduction of different means of copy protection which are still, unfortunately, necessary in Israel.

Even today, while large organizations do understand how important it is to use software legally, the average home user seldom buys any software, but prefers instead to copy from a friend, which is, of course, much cheaper. The phenomenon of unauthorised use of software provides viruses with a very fertile ground in which to spread and multiply.

Viruses 'At Home'

Israel, which is considered to be a technologically highly-developed country, is estimated to have PCs in over 40% of homes. Most of these machines fulfil multiple roles in education, entertainment and work.

Many users share data via diskettes or BBSs: it is not rare for these media to be virus-infected. This has the result that home users are very much aware of the virus threat. Many do not settle for just one anti-virus program; rather, they obtain several, 'to be on the safe side'. As these anti-virus programs are also often used illegally, and are not updated frequently enough, many viruses survive and replicate.

The infection of home computers by viruses also affects the corporate environment adversely. Many employees bring work home: once they return it to the workplace, there is a virus on the diskette which, within a space of hours, is able to infect the entire network.

Foreign Diseases

A large proportion of employees in Israeli high-tech industries are sent abroad temporarily for their work: they travel to client sites, installing or customising products. Many such sites are located in exotic countries with which Israel, for political reasons, trades. Many viruses have been introduced in Israel by employees returning home with infected diskettes imported from customer bases.

Today, in Israel's high-tech community, the *Internet* is becoming a vital working tool, similar to the telephone or the fax. Exchanging software through the *Internet*, and thus viruses, has become as easy as pressing a mouse button. Paradoxically, although many organizations have strict policies regulating software use, many allow free access to the *Internet*, enabling virus transmission via ftp or Email.

Infection by Companies

A recent development has been that large international software companies are sending software to be packaged in Israel. These multi-nationals use local sub-contractors, with little or no experience in large-quantity duplication, to duplicate floppy disks. There have been incidents in which internationally known companies shipped shrink-wrapped software with boot sector viruses, probably originating from infected master diskettes during duplication.

It is an interesting fact that, throughout the world, boot sector viruses are considered the most common infectors, as they can even travel on diskettes which contain only data. For some odd reason, in Israel, unlike Europe and the US, the 5.25-inch diskette is still more widely used than the 3.5-inch size. Approximately 80% of all diskettes sold are 5.25-inch: only 20% are 3.5-inch.

Since it is far more difficult to place a write-protect label on a 5.25-inch diskette than to open a notch on a 3.5-inch, users seldom bother with these labels. Even worse, most of the software houses selling 5.25-inch diskettes do not bother attaching write-protect labels in the first place.

There are several hundred BBSs in Israel, almost all of which are interconnected via networks or dial-up links, from which the latest software may be downloaded. In many cases, such programs are illegal and may even be hacked, copy-protected software. Some of them carry, and hence share, virus-infected samples.

Viruses in the Academic Community

Students in all Israeli universities had access to the *Internet* long before the corporate and high-tech industry. This was, and still is, a major gateway for shareware and freeware software - in addition, naturally, to the most recent viruses.

It is also not rare for students to 'lend' their *Internet* account details to friends, thus allowing any hacker to browse through the *Internet*, sending and receiving files at the expense of the academic community, and thus increasing threats to university information systems. Despite the resultant unsurprising damage, universities do little to regulate this virus superhighway into the country.

Almost every computer in schools, universities, and other accessible public places is virus-infested. Not only has the situation not improved since the onset of Jerusalem, it has deteriorated, due to the daily emergence of scores of new virus mutations.

Local Viruses

As elsewhere, Israel has its share of virus writers. Although the Jerusalem virus was first discovered here, it is not believed to have originated here. However, the first file stealth virus, and one of the best of its type, Frodo.Frodo.A (4K), was developed in Israel, its purpose to demonstrate that it is possible to overcome anti-virus programs.

Later, other and more famous viruses (including Haifa and Ariel) were developed, and quickly spread throughout Israel. We even had the Typo virus hacked and 'localized' to make it compatible with the Hebrew character keyboard. Only a handful of viruses have been developed in Israel, probably because local hackers have found more interesting ways to cause harm, like cracking copy-protected software!

Developing Integrity

Anti-virus software used in Israel is mainly of local origin - software utilizing the Hebrew language is a very important factor in buying decisions. Israel is known as an anti-virus country. Several of today's internationally-famed anti-virus programs were developed by Israeli companies.

Corporate users are usually aware of the virus problem, and many do their best to solve it. There are, however, companies which still refuse to deal with the issue. They cling to the ostrich theory that 'It will not happen to me', and operate by 'putting out local fires', ignoring larger issues.

Some companies think their network servers are well protected, and thus do not care what happens to individual workstations, feeling that it is the user's fault if a workstation is infected. Obviously, when a virus does infect, the damage sustained to the organization as a whole is huge.

An effective way to combat viruses was recently introduced here; a closed environment of diskette media. All computers in this environment have a program to read and write only authorised diskettes, reducing the possibility of introducing illegal software. The method has been accepted as standard by certain Israeli government ministries.

What the future will bring to Israel remains to be seen. One thing is certain: as elsewhere, viruses will remain, as will the battle against them.

PRODUCT REVIEW 1

The Doctor for NetWare

Jonathan Burchell
JC Designs

The *DOCTOR NLM* is a product from *Thompson Network Software*, based in Georgia, USA. Until 1994, the company was a subsidiary of the Australian *Leprechaun Software*.

The 'About TNS' appendix of the manual states that the company was founded by Roger Thompson, 'the creator of the world's first generic anti-virus product' and that it has been involved in anti-virus research since 1987. One presumes that a pedigree of this length implies an excellent product, so…

Presentation and Installation

The *DOCTOR NLM* comes on a single, high-density 3.5-inch floppy disk. The manual consists of 44 A5 single-sided pages bound into a ring binder, which sits inside a secondary plastic case. Although the manual has a very clean look and does a good job of explaining the installation procedure and various program options, it is extremely light on in-depth technical information.

This product will run on all servers running *NetWare v3.11* and *v3.12*, and on those running *NetWare v4.x*, provided that Bindery Emulation is enabled. Like most current *v3.x* NLMs, the software depends on having reasonably recent versions of such critical system components as NWSNUT and CLIB installed.

The manual identifies four critical NLMs, and asks the user to ensure that identical or later releases of each are installed. Strangely, the NLMs are identified by file date and not by version number. No details of how to update the NLMs are given - it seems a shame, considering there is space on the disk, that *Thompson* has not chosen to distribute a file server patch set.

Installation is performed by hand, and consists of copying the contents of the floppy disk (two files; the NLM itself and a signature database) to the server. The manual states that these should be copied to the SYSTEM directory, but I had no problem with installing them to alternative locations.

In addition to the installation and operation instructions, the manual contains a six-page chapter providing an overview 'About Viruses'. This is very well written, and should be developed further, to include greater detail on specific virus behaviour and practical information on dealing with an outbreak. The manual has no virus encyclopædia - perhaps a sensible move, considering how fast such things become outdated. However, not even the software lists viruses detected by the current database.

Administration and Operation

Once installed, the software is loaded simply by typing 'LOAD <PATH>\DOCNLM'. Without an install program, there is no automatic attempt made to load the NLM at AUTOEXEC.NCF time. Oddly, the manual makes no mention of the desirability of such a strategy.

No remote administration tools are provided: the NLM is configured and maintained either via the server console, or remotely using RCONSOLE. The software presents a standard *Novell*-type menuing interface, and the screen is divided roughly halfway between the current status/feedback area, and the configuration menuing area.

Main Menu

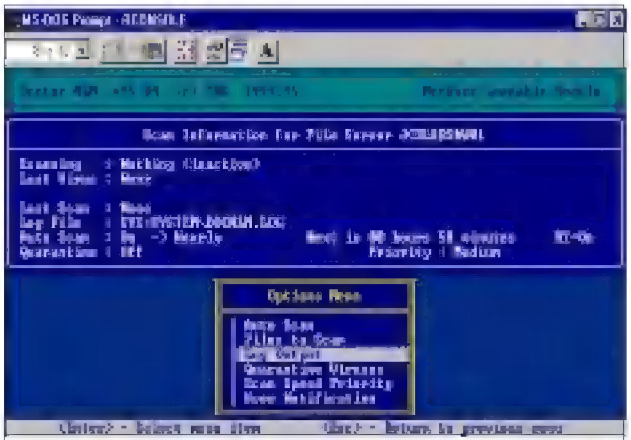
There are two principal options offered in the main menu: to perform an immediate scan, or to load new signatures. The immediate scan menu offers three sub-options, controlling which areas of the server are scanned. The choices are to scan the entire server, or a specified volume or path. Specifying a path automatically includes all sub-directories.

The 'load new signatures' option allows the NLM database to be updated whilst the NLM is running. Updates to the virus signature file are available electronically either from the *Thompson* BBS or by ftping to their *Internet* site.

Options Menu

The options menu offers AutoScan, which files to scan, log output, quarantine viruses, scan speed priority, and user notification. AutoScanning, or scheduled scanning, can be globally enabled or disabled. Once enabled, scan frequency may be selected: this can be hourly, every three, six, or twelve hours, or once a day at a specified time. No options to specify a day of the week or of the month are included.

It is also possible to specify paths to be scanned, via a list (which may contain up to 50 items) of Volume and Path names. Included under AutoScan is the option to check



The Options Menu of the *DOCTOR NLM* allows selection from several choices, according to exactly what is to be scanned.

'executables' in real-time as they are written to the server. No option exists to check files in real-time as they are read from the server.

The option on which files to scan allows two separate lists to be defined: a list of file types to scan specifically, and a list to exclude. The default include list is *.BIN, *.COM, *.EXE, *.OV?, & *.SYS (wild cards are allowed). The lists defined affect all scanning operations.

The *DOCTOR* can be instructed to log operation and virus identifications to two different places: its own log file (default name DOCNLM.LOG) and/or the system log file. No options are provided to view, filter or print these log files. The system log file can be viewed using SYSCON.

The quarantine viruses option allows specification of a path into which to move the infected file. The default name is SYS:\SYSTEM\INFECTED, which can be changed if desired. If quarantine is not selected, the files remain where they were discovered, unaltered in any way. It would be nice to see an option to delete or disable infected files automatically. It is also simplistic merely to move the infected file to a quarantine directory. A better approach would be to move it to a quarantine area and change its name: this might help identify where the file originated, and resolve conflicts in the event of multiple infections with the same name - this approach has already been adopted by other developers.

There is also an option on scan speed priority, which allows a trade-off between scan speed and server utilisation to be set. Three sub-options are provided; low, medium and high. These represent 10% to 40% (maximum) utilisation.

The option for user notification permits specification of a list of users to be notified on discovery of a virus. Only logged-on users will receive the notification, which is this non-customisable message:

```
DocNLM {SERVER NAME}: Found a virus !
```

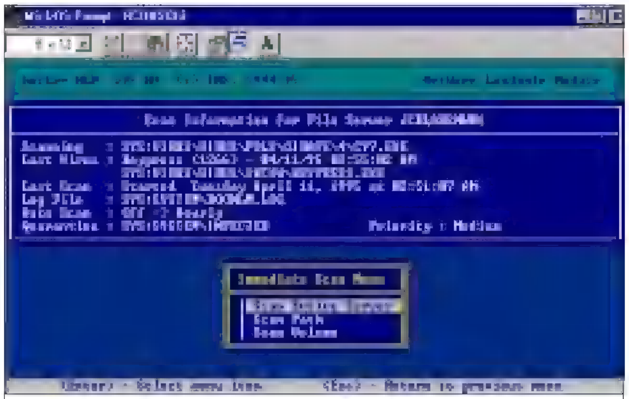
Real-time Scanning Performance Overhead

To establish the overhead of enabling real-time scanning, a directory of files was copied from a workstation with and without real-time checking enabled. The directory consisted of 231 real files, of which 57 were executables (totalling 5MB). The remainder were data files, the total data size was just over 10 MB.

Without real-time scanning, the copy took 69 seconds, and with real-time scanning enabled it took 74 seconds, an increase of 5 seconds or just 7.2%. This is a very creditable overhead figure, which hopefully will not increase dramatically when the detection ratios are improved.

Results

A major problem was discovered in use: the standard test procedure for background scanning is to copy all of the virus test-set to the server, with real-time checking disabled,



The Immediate Scan Menu allows a choice of where to scan.

and then to start off a background scan. As the test-set is now over 5,000 files in size, good sense demanded a coffee break at this point.

Once the scan has finished, a program is run which analyses the contents of the test-set directories, and works out which files have been put into the quarantine directory (or deleted). Every time this procedure was adopted with the *DOCTOR NLM* loaded, the file server hung whilst looking for viruses.

I double-checked that this happened with workstations connected/disconnected, and with quarantining and logging enabled/disabled. The hang happened in the POLY/ONE_HALF directory. Eventually, I solved the problem by presenting the virus directories to the scanner one by one. The hang still happened in the ONE_HALF directory, but I did get complete scans for all other directories.

The hang happened some way into the directory - the checker had shown no sensitivity to the virus prior to the hang, so I do not think that the test results are skewed, although a major hang such as this is rare to find these days, and as such is somewhat worrying.

The features offered by the *DOCTOR NLM* place it firmly in the small site, limited feature class. It would probably be suited to a single-server site, as no server to server features exist. A niche does exist for such products, providing they offer good detection and are priced accordingly.

Unfortunately, the detection offered by this version of the *DOCTOR NLM* is appalling, and currently offers little real protection. When contacted, *Thompson Network Software* agreed that rates could be improved; however, they stated that this version was one which they were 'working in'. The current release of the product, says *Thompson*, has much higher detection rates - on a par with its DOS scanner [*see VB DOS Scanner Comparative Review, July 1995, p.14*].

There cannot be any excuse for not getting 100% on the 'In the Wild' test-set, or a high score on the 'Standard' test-set. High scores here mostly rely on a complete signature database. It may be that the test results were skewed by the same problem that was causing the server to hang: when we resolve this issue we will publish an update.

DOCTOR NLM

Detection Results:

Main Scanner:

Standard Test-Set[1]	107/230	46.5%
In the Wild Test-Set[2]	87/126	69.0%
Polymorphic Test-Set[3]	114/4796	23.8%

Overhead of Real-time Scanning:

Time to copy 231 files

Without scanner	69 seconds
With scanner	74 seconds

(This is an increase of only 5 seconds, or 7.2%.)

Technical Details

Product: *The DOCTOR NLM*.

Developer: *Thompson Network Software*, 2619 Sandy Plaines Road, Marietta, Georgia 30066, USA.

Contact Numbers: Tel. +1 404 971 8900; Fax +1 404 971 8828; BBS +1 404 971 8886.

Price: US$300.00 per server per annum, with unlimited workstations. This includes access to their Bulletin Board System (see contact number above), unlimited upgrades (also via BBS), and technical support.

Hardware used: Client machine - 33 MHz 486, 200 Mbyte IDE drive, 16 Mbytes RAM. File server - 33 MHz 486, EISA bus, 32-bit caching disk controller, *NetWare 3.11*, 16 Mbytes RAM.

Each test-set contains genuine infections (in both COM and EXE format where appropriate) of the following viruses:

Standard Test-Set: As printed in *VB*, January 1994, p.19 (file infectors only).

In the Wild Test-Set: 4K (Frodo.Frodo.A), Barrotes.1310.A, BFD-451, Butterfly, Captain_Trips, Cascade.1701, Cascade.1704, CMOS1-T1, CMOS1-T2, Coffeeshop, Dark_Avenger.1800.A, Dark_Avenger.2100.DI.A, Dark_Avenger.Father, Datalock.920.A, Dir-II.A, DOSHunter, Eddie-2.A, Fax_Free.Topo, Fichv.2.1, Flip.2153.E, Green_Caterpillar.1575.A, Halloechen.A, Helloween.1376, Hidenowt, HLLC.Even_Beeper.A, Jerusalem.1808.Standard, Jerusalem.Anticad, Jerusalem.PcVrsDs, Jerusalem.Zerotime.Australian.A, Keypress.1232.A, Liberty.2857.D, Maltese_Amoeba, Necros, No_Frills.843, No_Frills.Dudley, Nomenklatura, Nothing, Nov_17th.855.A, Npox.963.A, Old_Yankee.1, Old_Yankee.2, Pitch, Piter.A, Power_Pump.1, Revenge, Screaming_Fist.II.696, Satanbug, SBC, Sibel_Sheep, Spanish_Telecom, Spanz, Starship, SVC.3103.A, Syslock.Macho, Tequila, Todor, Tremor (5), Vacsina.Penza.700, Vacsina.TP.5.A, Vienna.627.A, Vienna.648.A, Vienna.W-13.534.A, Vienna.W-13.507.B, Virdem.1336.English, Warrior, Whale, XPEH.4928.

Polymorphic Test-Set: 4796 genuine samples of: Cruncher (25), Uruguay.4 (75), Satanbug (100), Girafe (1024), Mutation_Engine (500), One_Half (1024), Pathogen (1024), Smeg_03 (1024).

PRODUCT REVIEW 2

RingFence

Dr Keith Jackson

RingFence, from *S&S International*, is a very different product from the scanners which are usually discussed in *VB* reviews: it attempts to prevent viruses from having any effect by forming a barrier to prevent their introduction.

RingFence prevents *any* unauthorised floppy disk from being accessed; therefore, as long as the person authorising diskettes takes due care to inspect/check their content, it should be impossible for a virus to enter via that route. This is summarised in the manual, which states that *RingFence* 'is a program that prevents your computer from accessing unauthorised diskettes'.

The product was last reviewed by *VB* in July 1993: as it has recently had a major upgrade, a revisit seemed appropriate. In this review, I intend to explain the *RingFence* features, and measure the overhead it introduces when present.

Features

As well as monitoring floppy disk access, *RingFence* can encrypt (and/or decrypt) the entire contents of a floppy disk. A lower level of file security is offered by a feature denoted as 'Padlocking'. Both of these features are discussed below.

Access to serial and parallel ports can also be made available selectively. Controls provided can be used to prevent users loading files onto a PC through the serial line; e.g. via a modem, or through devices connected to the parallel port.

Without introducing extra hardware, it is impossible to prevent any security system which uses programs stored on a hard disk from being bypassed by a floppy disk boot. However, *RingFence* manipulates the hard disk's boot sector and/or partition sector to ensure that, even if a floppy disk boot is performed, the hard disk cannot subsequently be accessed. *RingFence* also prevents writes to the hard disk boot sector, the hard disk partition sector, or the files CONFIG.SYS and AUTOEXEC.BAT.

If 'any attempt is made to tamper with the RingFence programs on your hard disk', the documentation claims that the product makes the hard disk read-only. This can only be partially true, as there must be a level of alteration beyond which the product's features are hopelessly compromised.

Installation

RingFence was provided on a single 1.44 MB floppy disk, which came with a manual (see below) and various pieces of bumph. To install *RingFence*, it is necessary to provide a write-protected DOS boot disk, and a blank floppy, for every PC on which *RingFence* is to be installed. The two files copied to the blank floppy during installation enable it to be used as a recovery disk should disaster strike.

Installation is simply a matter of typing INSTALL, making a recovery disk, confirming that the PC has been fully backed up (answering negatively terminates installation), and providing the name of the subdirectory into which the product's files should be installed.

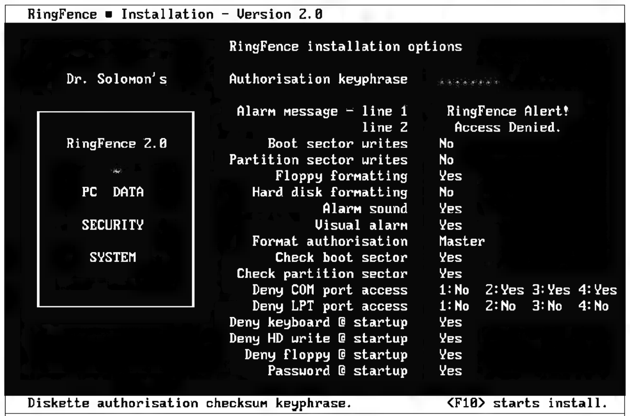
Previous versions would install several files in the root directory of the hard disk, a practice I abhor. This latest version only leaves two copies of the boot sector and the partition sector (as hidden files) in the root directory. *RingFence* now keeps its files tucked away in its own subdirectory: this is a step forward.

Both the supervisor and the power-up passwords must be entered twice during installation. The supervisor password is required to execute *RingFence's* control program (RFMASTER), and to de-install *RingFence*. The power-up password must be entered every time the PC is rebooted with *RingFence* active. The installation program alters AUTOEXEC.BAT to ensure that the *RingFence* software is launched when the PC is rebooted from the hard disk.

De-installation

Being a cautious soul, I always try to de-install software before reviewing it. With *RingFence*, it is necessary to boot from a write-protected DOS disk, and execute a program called DEINSTALL from the *RingFence* master floppy disk.

The floppy disk reboot is mandatory. If omitted, DEINSTALL warns: 'There are protections active in memory. Reboot from a clean DOS disk', and terminates. After successfully executing DEINSTALL, an onscreen message states that Stage 1 is complete, and the PC should



```
RingFence ■ Installation - Version 2.0

                    RingFence installation options

 Dr. Solomon's      Authorisation keyphrase    ********

                     Alarm message - line 1     RingFence Alert!
 RingFence 2.0                       line 2      Access Denied.
                         Boot sector writes    No
                     Partition sector writes   No
 PC  DATA                 Floppy formatting    Yes
                       Hard disk formatting    No
 SECURITY                       Alarm sound    Yes
                               Visual alarm    Yes
 SYSTEM                Format authorisation    Master
                          Check boot sector    Yes
                     Check partition sector    Yes
                       Deny COM port access    1:No  2:Yes 3:Yes 4:Yes
                       Deny LPT port access    1:No  2:No  3:No  4:No
                    Deny keyboard @ startup    Yes
                    Deny HD write @ startup    Yes
                      Deny floppy @ startup    Yes
                        Password @ startup     Yes

Diskette authorisation checksum keyphrase.              <F10> starts install.
```

The installation program will alter AUTOEXEC.BAT to ensure that, once installed, the *RingFence* software is always launched when the PC is rebooted from hard disk.

be rebooted and DEINSTALL re-executed. This sounds simple, but fell in a heap with 'RingFence not installed in this system. De-installation failed'. Eeek! And it's not lying: reboot from the hard disk and *RingFence* is still there.

Given that *RingFence* changes the boot sector and the partition sector of the hard disk, I resisted the temptation to delete all files belonging to the product - that could result in my PC being rendered completely inaccessible. The manual explained that, if a system failure is encountered, a program called RFRESTOR is provided which should get things going again. I tried this, but to no avail. *RingFence* was still there, though the de-installation program could not see it!

Having tried to de-install several times, I caved in and called the developers for help. After discussion, they thought my problems were caused because the attributes of *RingFence's* memory-resident program were set to read-only and hidden, with the result that the de-installation program was refusing to delete this file. This was stated to be a known bug.

I used the *MS-DOS* ATTRIB command to remove these attributes, and found to my surprise that they had not changed. After several retries, and more discussion with technical support, it dawned that this was caused by *RingFence* resetting the attributes immediately after I had altered them. This program definitely did not want to let go!

I had not installed *RingFence* in supervisor mode, so very few files had been installed on my hard disk. However, technical support assured me that the program(s) needed to complete the de-install were on the *RingFence* master floppy disk, which I duly tried to access.

RingFence was still installed, and would not permit accessing the master diskette, as this was not an authorised disk. The program which could be used to authorise floppy disks was only stored on the inaccessible *RingFence* master floppy disk, and would not authorise anything if it was executed on a PC without *RingFence* running. Stalemate.

Senior technical support personnel were consulted: they were sure that although the onscreen messages said otherwise, *RingFence* probably had been removed from the boot sector and the partition sector.

If, I was told, I booted from floppy disk, edited the file AUTOEXEC.BAT to remove *RingFence* manually, then rebooted from the hard disk, all should be OK. Probably? Should? If they were wrong, I could be looking at having to do some extensive data recovery.

Taking a deep breath, I followed their advice. It worked. I could access the hard disk, use ATTRIB to remove the read-only and hidden attributes of *RingFence's* memory-resident program, and manually delete this file. Phew - *RingFence* had finally gone.

In spite of all these shenanigans, I remained impressed with the help I received from the *RingFence* technical support personnel. Thanks, Eamonn! Particularly impressive was the way that technical support had obviously been told to refer to others whenever they felt that they had reached the limit of their knowledge, and not to blunder on regardless. That being said, they did know that I was writing a review for *Virus Bulletin*.

I *needed* technical support to de-install *RingFence*. Not an auspicious omen. I have omitted much detail from this saga, and concentrated on the main problems, but all in all it took almost an entire morning before I succeeded in de-installing *RingFence*. God knows what naïve users would do.

I was left with several questions about de-installing the product. Why did it leave behind a file called AUTOEXEC.O? - nobody seemed to know. Why are the text messages in the de-installation program misleading? If *RingFence* was removed from the boot sector and the partition sector, why not delete the *RingFence* files anyway? Why is it not mentioned in the documentation that the memory-resident program resets its attributes, if altered? I could go on.

[*Editor's note: S&S International is aware of the problem, and is currently testing a fix which will be sent to all registered customers as soon as testing is complete.*]

"RingFence … attempts to prevent viruses from having any effect by forming a barrier to prevent their introduction"

After all this, I reinstalled *RingFence*. When installation was complete, I tested the UPDATE facility of the installation program by trying to install *RingFence* again. This would not work. I could not even get the installation program to accept the name of the subdirectory where the *RingFence* files were located on the hard disk, and could find no information anywhere to tell me why this should be.

However, the restore disk was updated before this sticking point was reached. So how could I restore the original boot sector and partition sector if a system failure occurs? Just how should I update *RingFence*? The manual gives no real help on this matter.

I gave up trying, de-installed, and installed again in supervisor mode. This copied all of *RingFence's* files onto my hard disk, which hopefully would let me avoid a repeat performance of the above problems.

Supervisor Operation

RingFence must be controlled by a supervisor, whose job it is to validate diskettes, install/de-install *RingFence*, and introduce encryption as necessary. All supervisory functions are available through a program called RFMASTER. Unless the product has been previously installed on a supervisor's hard disk, it is necessary to authorise the *RingFence* master diskette to gain access to the program. Having to alter a master diskette in any way is not acceptable. Such beasts should be left in their original condition.

If a virus is present when the *RingFence* supervisor authorises a diskette, it can spread in the same manner as if *RingFence* were not present. Use of the scanner in *Dr. Solomon's Anti-Virus Toolkit* (from the same developer) is recommended, but a program is provided (RFLAUNCH) which can integrate any scanner with *RingFence*.

Documentation

The *RingFence* documentation comprises a 54-page, A5 volume which provides only a high-level description of the available functionality. The manual contains a decent Table of Contents, Glossary and Index. Installation and de-installation occupy most of the manual.

Unfortunately, the depth of detail provided in the manual is close to nil. In particular, users are given no indication of how 'Padlocking' operates (I assume that it rewrites the File Allocation Table in a non-standard manner). They are told that padlocking is 'a little less secure than encryption', which is untrue (it is much less secure) and unhelpful.

When explaining how the hard disk is protected following a floppy disk boot, the manual says cryptically: 'most low-level programs and disk utilities will be unable to access the hard disk'. Even given the disk editor provided in something like *Norton Utilities*? *RingFence* claims to work with 'all the leading proprietary disk compression programs', but does not give the names of those against which it has been tested. Useless, completely useless.

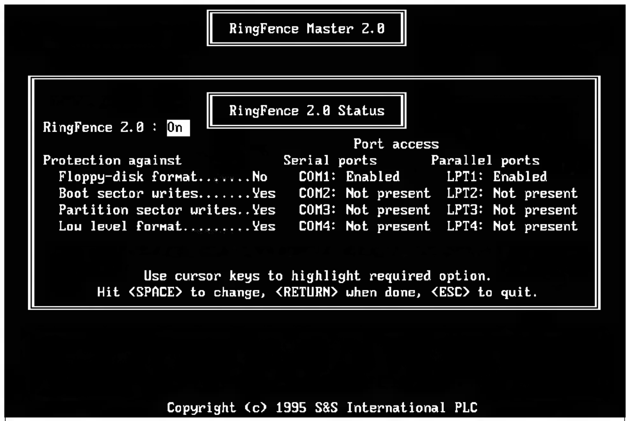
I could go on listing examples, but those above should make my point. Frankly, users deserve more detailed explanation of the various methods of operation. The manual reads as if it has been 'technically authored' (written but not understood!). New users will probably only look at it once, finding that they outgrow its level of usefulness on day one.

The manual discusses what to do with software packages which arrive on permanently write-protected floppy disks (which cannot be authorised as they cannot be written to). If it is not desirable to write to master disks (see above discussion), then it is necessary to turn *RingFence* off temporarily, install the software, and then re-activate *RingFence*. All this is a pain, and makes *RingFence* more suitable for stable systems than development systems.

Memory-resident Program

Much of *RingFence's* functionality is provided through a memory-resident program called RING.COM, stored as a hidden read-only file on the hard disk. I'm at a bit of a loss to understand why it has been given hidden attributes.

Why bother? A quick perusal of AUTOEXEC.BAT will reveal to anyone that such a file must exist, and even discloses the subdirectory location. So why bother playing games with file attributes? It seems pointless, and as discussed with respect to de-installation, may also cause severe problems.



```
                    RingFence Master 2.0

                    RingFence 2.0 Status
RingFence 2.0 : On
                                      Port access
Protection against              Serial ports       Parallel ports
  Floppy-disk format.......No     COM1: Enabled      LPT1: Enabled
  Boot sector writes.......Yes    COM2: Not present  LPT2: Not present
  Partition sector writes..Yes    COM3: Not present  LPT3: Not present
  Low level format.........Yes    COM4: Not present  LPT4: Not present


           Use cursor keys to highlight required option.
      Hit <SPACE> to change, <RETURN> when done, <ESC> to quit.



             Copyright (c) 1995 S&S International PLC
```

Once *RingFence* is enabled, the user is informed of its status, and is allowed to alter individual options.

RingFence occupies 5.5 Kbytes when installed in memory, and provides its security function by hooking into four DOS interrupts (including Ints 0Bh, 0Ch, 14h and 17h).

The overhead introduced by *RingFence* was measured by timing how long it took to copy 25 files (694 Kbytes) from a subdirectory on the hard disk of my test PC to the root directory of a 3.5-inch, 720 Kbyte floppy disk.

Without *RingFence*, these files could be copied in 1 minute 43 seconds. This rose to 3 minutes 55 seconds when *RingFence* was installed, and further to a whopping 5 minutes 14 seconds when the test files were copied to a floppy disk encrypted by *RingFence*.

A similar drastic overhead was measured when the 40 test files were deleted from the floppy disk. Without *RingFence*, the deletion took just 7 seconds. With *RingFence* present this rose to 1 minute 32 seconds, and introducing encryption further increased the deletion time to 2 minutes 27 seconds.

Just think of it - nearly two and a half minutes to remove a small group of files stored on a floppy disk. Is that acceptable? Probably not.

Encryption

RingFence can encrypt the entire contents of a floppy, but not a hard, disk. Using the DOS version of RFMASTER took 4 minutes 18 seconds to encrypt a 3.5-inch, 720 Kbyte floppy disk. This time remained the same, whether the floppy disk was empty or full, and whether or not different types of authorisation, 'master' or 'area', were in effect.

Decryption times were always identical to encryption times. Rather surprisingly, the *Windows* version of RFMASTER needed only 1 minute 53 seconds to carry out the same encryption task. Is this the first program ever where a task is immeasurably faster under *Windows* than under DOS?

The encryption software was designed to ensure that, once a disk had been encrypted, further encryption was impossible. Similar interlocking applied to all modes of authorisation. No details are provided in the documentation as to what type of encryption is used, which makes it impossible to judge how secure the encryption algorithm actually is. Users deserve more than this paucity of detail.

The Rest

Whenever *RingFence* detected an attempt to access an unauthorised floppy disk, it produced a warbling noise as a warning, and if the screen was in a compatible mode, displayed a message stating: 'RingFence alert, Access denied' in the middle of the screen. The content of this message can be tailored by the supervisor.

In supervisor mode, the program RFMASTER allows various features to be enabled, disabled or amended. For instance, a floppy disk can be 'padlocked', an operation which takes place almost instantaneously on any floppy disk, under either DOS or *Windows*.

Similarly, writing to the hard disk boot sector and/or partition sector, disk formatting, serial port access, parallel port access can all be either enabled or disabled. It is even possible to disable the keyboard and/or the ability to write to the hard disk during a reboot.

There are definite problems associated with *RingFence's* control of formatting. Even though an onscreen message stated 'Floppy disk format protection off', it still proved impossible to format a floppy disk. *Norton's* Safe Format program failed after displaying a *RingFence* error message. When the *MS-DOS* command FORMAT was used, the format program simply replied: 'Checking existing disk format', and then hung. The only cure was a reboot.

If the floppy disk was removed from the drive after the format program had hung, the disk drive continued to make a noise like an outboard motor. This was probably caused by the head banging up against the end-stop, which cannot do the heads any good.

The same features are provided whether or not the DOS or *Windows* version of RFMASTER is used. However, the *Windows* version does exhibit quirks and one severe problem. The quirks relate to the action performed by using keystrokes to move around rather than the mouse. For instance, when in the main selection screen of RFMASTER, the down arrow key moves one icon to the right - this is a thoroughly confusing action.

It proved impossible to execute RFMASTER more than once if the encryption facility was used. When the second execution commenced, RFMASTER immediately fell over and produced a GPF error. This problem was repeatable, did not occur unless the floppy disk encryption facility was invoked, and could be reset by rebooting the PC (whereupon it again happened on the second execution of RFMASTER).

Conclusions

Overall I quite like *RingFence*, but its de-installation really does need sorting out - with a big stick! There are non-trivial bugs declared in the README file, and, as I found out the hard way, there are also other bugs which remain lurking to trap the unwary.

In similar fashion, I was disappointed by the bugs which I encountered whilst trying to format a floppy disk, and whilst using RFMASTER under *Windows*. It really is not acceptable for a 'tested' piece of software to fail immediately (and repeatedly) if it is executed more than once.

Overall, I was saddened by the bugs which are apparently still present in *RingFence*. Was it really ready for a version 2.0 launch, or was the product shoved out of the door when the chosen date arrived? I suspect the latter.

RingFence offers features which provide an effective barrier to prevent the introduction of any unwanted (that is to say, unauthorised) floppy disk. However, if you are considering purchasing *RingFence* - for instance, if you supervise a location where it is important that floppy disk access is controlled on all PCs - then I strongly recommend looking very carefully at the overhead which is imposed on floppy disk operation.

If floppy disks are used frequently, you may well find that the disk access slowdown is unacceptable. I must admit that I was surprised by just how much of an overhead is imposed by *RingFence*.

All this gets in the way of a product which performs its core task well. The developers would be well advised to look to their technical laurels and sort out the bugs in *RingFence*, rather than (as has been evident recently) appearing to be concentrating on the marketing of their software packages.

Technical Details

Product: *RingFence.*

Version evaluated: 2.00D.

Serial number: None visible.

Developer/Vendor: *S&S International*, Alton House, Gatehouse Way, Aylesbury, Buckinghamshire HP19 3XU, UK, Tel: +44 1296 318700; Fax: +44 1296 318777, email: sales@sands.co.uk; Compuserve: GO DRSOLOMON.

Price: Single-user licence, £39.95 + VAT + £5.50 p&p; 5-100 users, £20.00 per PC; 101-250 users, £15.00 per PC; 251-100 users, £12.00 per PC; 501 - 1000 users, £9.00 per PC. All prices are per annum and include full technical support.

Availability: *RingFence* requires an AT-compatible PC running at least *MS-DOS v3.x* (or equivalents *Novell DOS* or *IBM DOS*). *RingFence* occupies approximately 200 Kbytes of hard disk space; 5.5 KB of memory. When supervisor features are invoked, it also requires at least a 386/33 processor, 640 Kbytes of RAM and 1.2 Mbytes of hard disk space.

Hardware used: *Toshiba 3100SX*, a 16MHz 386 laptop, with 5 Mbytes of RAM, a 3.5-inch (1.44M) floppy disk drive, and a 40 Mbyte hard disk, running under *MS-DOS v5.0.*

SUBSCRIPTION RATES

Subscription price for 1 year (12 issues) including first-class/airmail delivery:

UK £195, Europe £225, International £245 (US$395)

Editorial enquiries, subscription enquiries, orders and payments:

Virus Bulletin Ltd, 21 The Quadrant, Abingdon, Oxfordshire, OX14 3YS, England

Tel 01235 555139; International Tel +44 1235 555139
Fax 01235 531889; International Fax +44 1235 531889
Email editorial@virusbtn.com
CompuServe address: 100070,1340

US subscriptions only:

June Jordan, *Virus Bulletin*, 590 Danbury Road, Ridgefield, CT 06877, USA

Tel +1 203 431 8720; Fax +1 203 431 8165

END NOTES AND NEWS

Reflex Magnetics launched a product codenamed *Hunter* at the *Networks '95* show in June. *Hunter* provides **five levels of virus protection on *Windows 95* systems**, and includes generic code emulation technology to look 'through' polymorphic encryption. It will go on general release later this summer. *Reflex* can be contacted on Tel +44 171 372 6666, Fax +44 171 372 2507.

***Compsec 95* will take place in London**, UK, from 25-27 October 1995. For details on the conference, contact Sharron Emsley at *Elsevier Advanced Technology* on Tel +44 1865 843721, Fax +44 1865 843958; email s.emsley@elsevier.co.uk.

Command Software Systems, Inc. has released ***F-PROT Professional 2.18***. The system provides a VxD for *Windows*, in addition to scanners for DOS and *Windows*, and a TSR for DOS. It is available now at a RRP of £90 from Command on Tel +44 171 259 5710, Fax +44 171 259 5753, email command@command.co.uk.

The fifth annual *Virus Bulletin* conference, ***VB '95*, will be held at the *Park Plaza Hotel* in Boston, Massachussetts**, from 20-22 September 1995. Internationally renowned virus and security experts will address the problems of virus protection in the 1990s. For more information, contact Petra Duffield, Conference Manager, on Tel +44 1235 555139, Fax +44 1235 531889.

InfoSecurity News announced the winners of its first **Readers Trust Awards** at the *LAN/SEC* conference in Washington at the end of June. Their 'best anti-virus product' award went to *Norton Anti-Virus*, from *Symantec Corporation*.

Information Security on the Internet is a **two day conference** taking place at the *Cumberland Hotel* in London, UK, on 25/26 September 1995, with post-conference workshops on 27 September. Tel +44 181 332 1112, Fax +44 181 332 1191 for information.

Both *Sophos* and *S&S International* have launched **World Wide Web sites**. The sites are reachable at http://www.sophos.com/ and http://www.sands.com/ respectively.

New Security Issues 1995 is a conference taking place at the *London Hilton Hotel* in London, UK, on 12/13 September 1995. The conference is subtitled 'Safeguarding New Technologies, Fraud and Enterprise Systems'. Information is available from Dipti Chauhan or Lisa Minoprio at *IBC Technical Services Ltd*, Tel +44 171 637 4383, Fax +44 171 636 1976.

Selfridge Thistle Hotel in London, UK, hosts the ***Information Security Managers Symposium* from 19-21 September 1995**, with an optional workshop on 18 September. Information is available from Louise Thomson, *Euromoney Publications Plc*, Tel +44 171 779 8944, Fax +44 171 779 8795.

The ***22nd Annual Computer Security Conference and Exhibition*** will be held in Washington, DC from 6-8 November 1995. The conference will feature over 120 sessions on various topics. Further information is available from the *Computer Security Institute* on Tel +1 415 905 2626, Fax +1 415 905 2626.

In Wallingford, Connecticut, USA, **disks infected with an unnamed destructive computer virus** were found on the desk of a former employee who, days before, had shot his manager in the leg before killing himself. Charles Coats was fired from *Corometrics Medical Systems*, which makes fœtal monitoring equipment on 24 May 1995, reports *Security Director's Digest*.

Reflex Magnetics will be presenting **two courses on computer security** - *PC Security* on 7/8 August 1995, and *Auditing PC Security* on 9/10 August 1995. Further information is available from Rae Sutton at *Reflex* (see contact numbers above).